Thoughts on parallelisation of initialisation phase and open issues

Vladimir Ivantchenko

28th Geant4 Collaboration Meeting

25–29 Sept 2023 Hokkaido University, Japan

Outline

- Use cases and the problem
- Current initialization of physics
- Improvements for Geant4 11.2
- Comments for parallelization of initialization

The problem and use cases

- If Geant4 is used in supercomputers with many nodes the longest initialization is performed in the master thread
  - All other threads should wait to start
- Traditional Geant4 simulation
  - May be sequential or muti-treaded
  - Physics is initialized when geometry is fully initialized
    - List of particle, list of elements and materials are already defined
    - In Geant4 tests often many runs are executed in one job, materials are changed between runs
  - It is the main use case tested within Geant4 everyday
- Geant4e – backward propagation of particles
  - Used, at least, in CMS
  - Geant4 geometry and magnetic field are initialized
  - RunManager is not used – initialization of EM physics is triggered by Geant4e
    - There is no master thread
    - There is no hadronic physics and no cuts
- Usage of only limited number of components of Geant4
  - Unit tests of Geant4 – RunManager is not used, usually sequential
  - In CMS FastSim may use FTFP model as an alternative – no geometry, no RunManager

Initialisation of EM physics

- Initialization of EM physics
  - PreparePhysicsTable(const G4ParticleDefinition&)
  - BuildPhysicsTable(const G4ParticleDefinition&)
    - Two methods are needed to retrieve tables from data files or alternatively to build from scratch
  - Each EM process triggers initialization of models belong to this process
  - Data structure may be created per material_cut_couple, or per element, or per material
  - Some processes compute numerical integrals, another compute cross sections and other values using formulas and/or G4LEDATA
    - We usually upload of data for materials/elements used in geometry
    - For a new run in the same job, we usually increment tables, not build from scratch
      - If extra material is added the previous is not deleted between runs
- In EM physics several processes used by each particle
  - Processes may be shared between particles
  - Models may be shared between particles/processes
    - In order do not overdo the same work data structures are filled for the 1st particle
- Energy loss processes for a given particle are combined
  - dEdx, range, and inverse range tables are prepared
- Process and model EM data are shared between threads
  - Normally fully initialized in the master thread but may be use cases when lazy initialization required
    - Because of a probability of lazy initialization other threads are locked during initialization

Initialisation of hadronic physics

- Initialization of hadronic physics
  - PreparePhysicsTable(const G4ParticleDefinition&) used minimally
  - BuildPhysicsTable(const G4ParticleDefinition&) is the main initialization method
    - Triggered initialization of models belong to the process
    - Triggered initialization of cross sections belong to the processes
    - In some models, significant data structures are filled
    - Models and cross sections may be shared between processes
- Data are retrieved from data tables or computed
  - G4PARTICLEXSDATA – uploaded for all atoms involved in run, shared between threads
  - In the current G4NEUTRONHPDATA lazy initialization is used
    - In the new approach, which is in progress the same method as in G4PARTICLEXSDATA will be used, at least, for cross sections
- Nuclear level data may be uploaded at initialization
  - Max atomic number can be defined (the default Z=30)
    - This is needed to avoid uploading of all high-Z elements data if not needed
  - Lazy initialization for remaining data
- Lazy initialization for radioactive decay data
  - This data belong to projectile and not to target, so not possible to know in advance what to initialize
- When any data file is uploaded all other threads are locked
  - Data structure may be per element and/or per isotope

Improvements for Geant4 11.2

- Since 10.X series of Geant4 the main method to provide initialization of shared data was to use a check if it is master or worker
  - IsMaster() method in EM classes
  - This is OK for “normal” use of Geant4 physics
  - Not working if master thread is not used
- Bug report #2546
  - Was confirmed also in G4HepEm
  - Order of initialization of static data may lead to crash
  - Extra model object instantiated in master thread may destroy initialisation
- In 11.1 a new approach was introduced
  - IsFirst check – if some data is not filled this class get flag isFirst
- In 11.2 more clean approach is proposed (like one used in CMSSW)
  - static std::once_flag applyOnce
  - ……..
  - std::call_once(applyOnce, [this] () { isInitilyzer = true; });
  - This construction defines that given class object is responsible to initialize static data
  - It is important that data structure does not delete in a model destructor but centrally end of job

Is it possible to parallelize initialization?

- The 1st answer: likely impossible
  - It is complicate to make in a simple but clean way
  - Different EM physics classes share data
  - Different hadronic classes also share data
  - Data structures not always known in advance, because uploaded data size is not known in all cases
  - Better not destroying Geant4
- It seems that substitution of isMaster() checks by other constructions is the first task for both EM and hadronics
  - This would prevent from situations like one described in #2546
  - It would be useful for several use cases
  - Does not introduce parallel initialisation

How to parallelize initialization?

- The 2nd answer: it is not completely excluded
  - It is possible to allocate memory for data structures for some processes/models before the start of computation of cross sections and probabilities
    - By material_cuts_couple in EM
    - By elements and/or isotopes for hadronic
  - The memory for uploaded data from files not known in advance in all cases
    - There may be different files even user custom file
    - It seems to be much more difficult to perform parallel upload and allocate memory efficiently
  - Computations for given Z, A, or material_cut_couple are independent, so potentially may be performed in parallel
    - It is possible if a loop may be parallelized
    - This may be introduced in models one by one where possible/necessary

No conclusion – the question is opened